Data Mining Lab 2020: Variational Fair Autoencoder

Intermediate Results

Mohammad Abir, Fabian Beringer

VFAE Short Refresher

- encodes features into latent space **z** using NNs
- naturally disentangles *sensitive variable* **s** from **x** by modeling them as independent
- also injects label information about **y**

  -> keeps **z** useful when predicting **y** but removes **s**

Dataset: Adult Income

Dataset

- Each instance **x** represents a person
- Labels **y**: whether the person has >50k annual income
- The protected attribute **s** is **gender:**

  67% Male, **33%** Female (protected class)
- **45,222** instances, **15** features each

Dataset

For pre-processing we:

- removed NaN’s
- binarized each feature using one-hot encoding
- bucketized continuous features into **5** buckets each
- resulted in **117** binary features

Challenges

Paper Omitting Details

- pre-processing not explained in detail
  - we followed another paper that was mentioned as reference
- no info about total training times
  - we trained for 100 epochs at max
- penalty scaling parameter **β** unclear for some datasets:

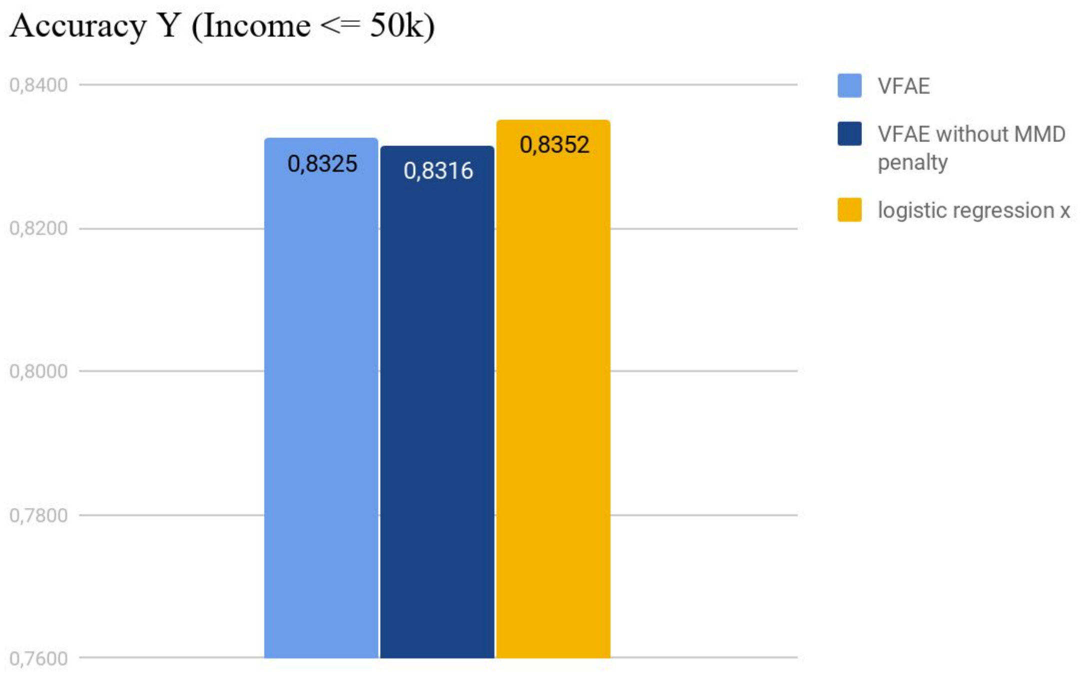
  “scaled according to validation set”
  - we found that **β=1.0** i.e. not scaling at all worked best

Further Challenges

- rather complex objective function, harder to debug
- non standard metrics i.e. discrimination have some specialties when it comes to implementation details
- designing a training pipeline for different models which depend on each other can be a bit tricky

Comparing Results

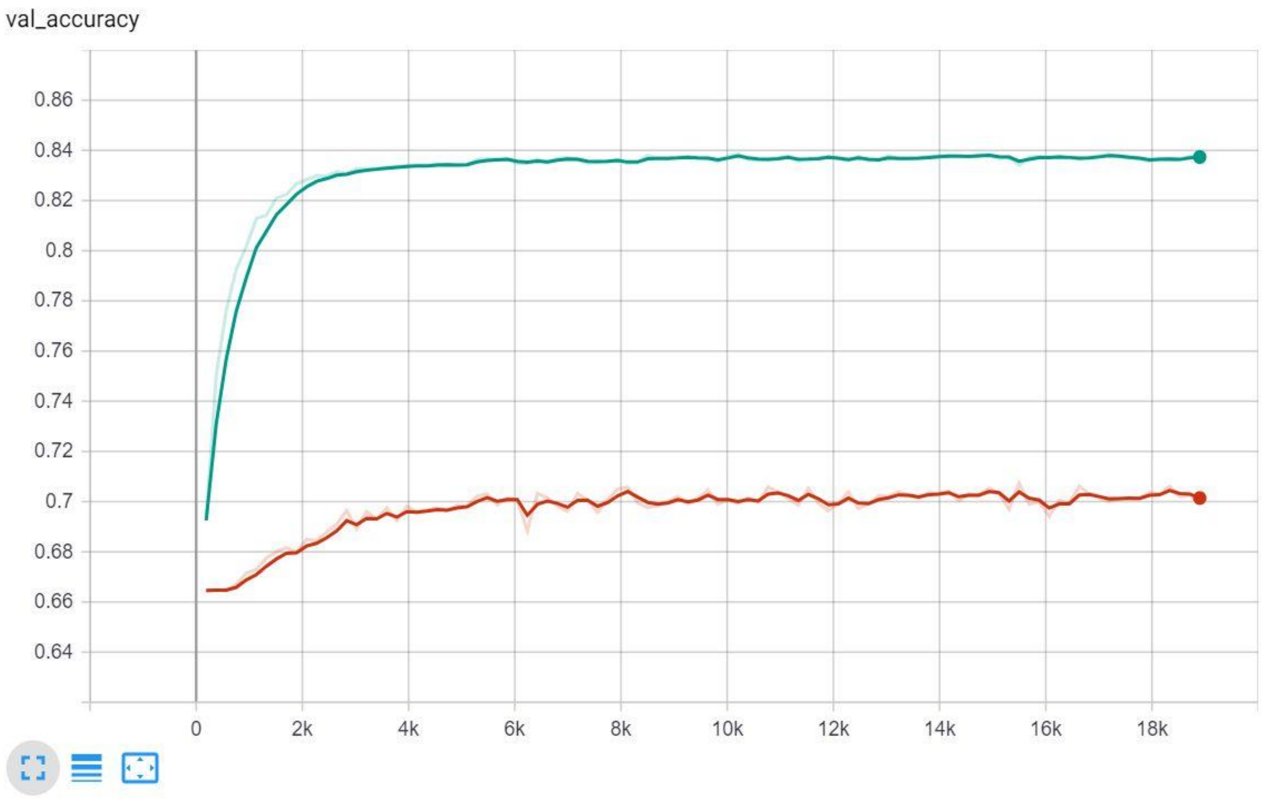
Accuracy on **y**, trained on **x** vs. **z**



Accuracy on **s**

Validation accuracy during training when predicting gender.

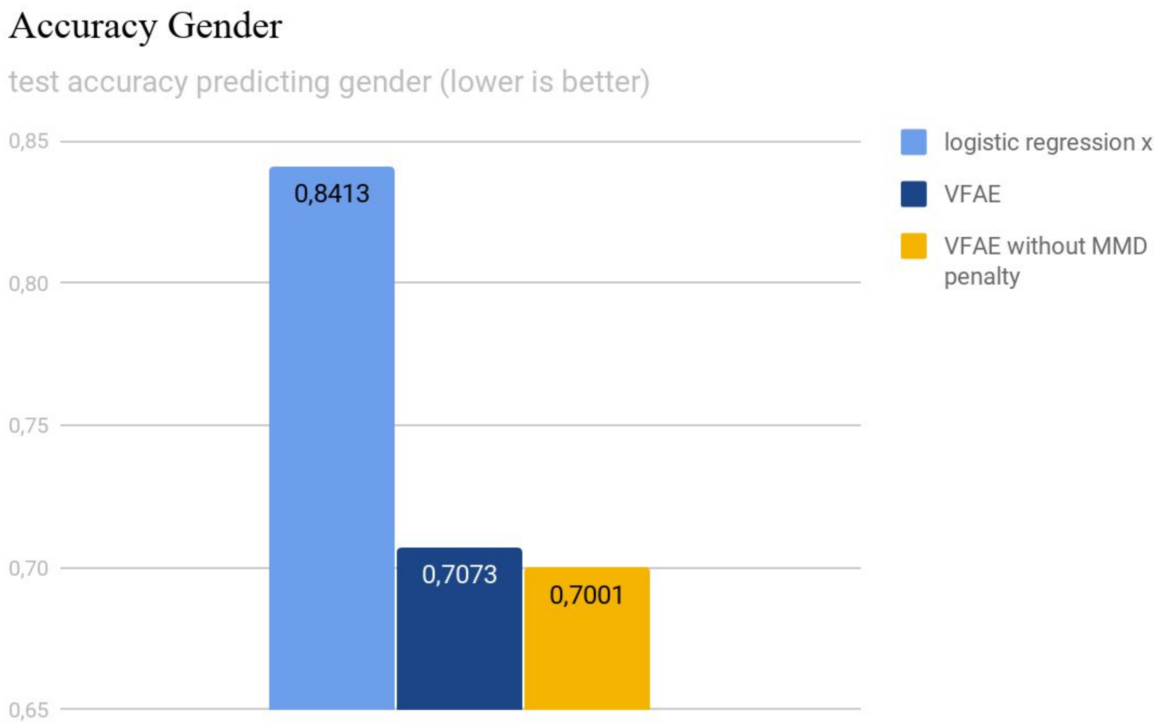
- slightly higher than mentioned in the paper **->** due to longer training time?



original features **x**

representation from VFAE **z**

Accuracy on **s**, trained on **x** vs. **z**



Discrimination Score

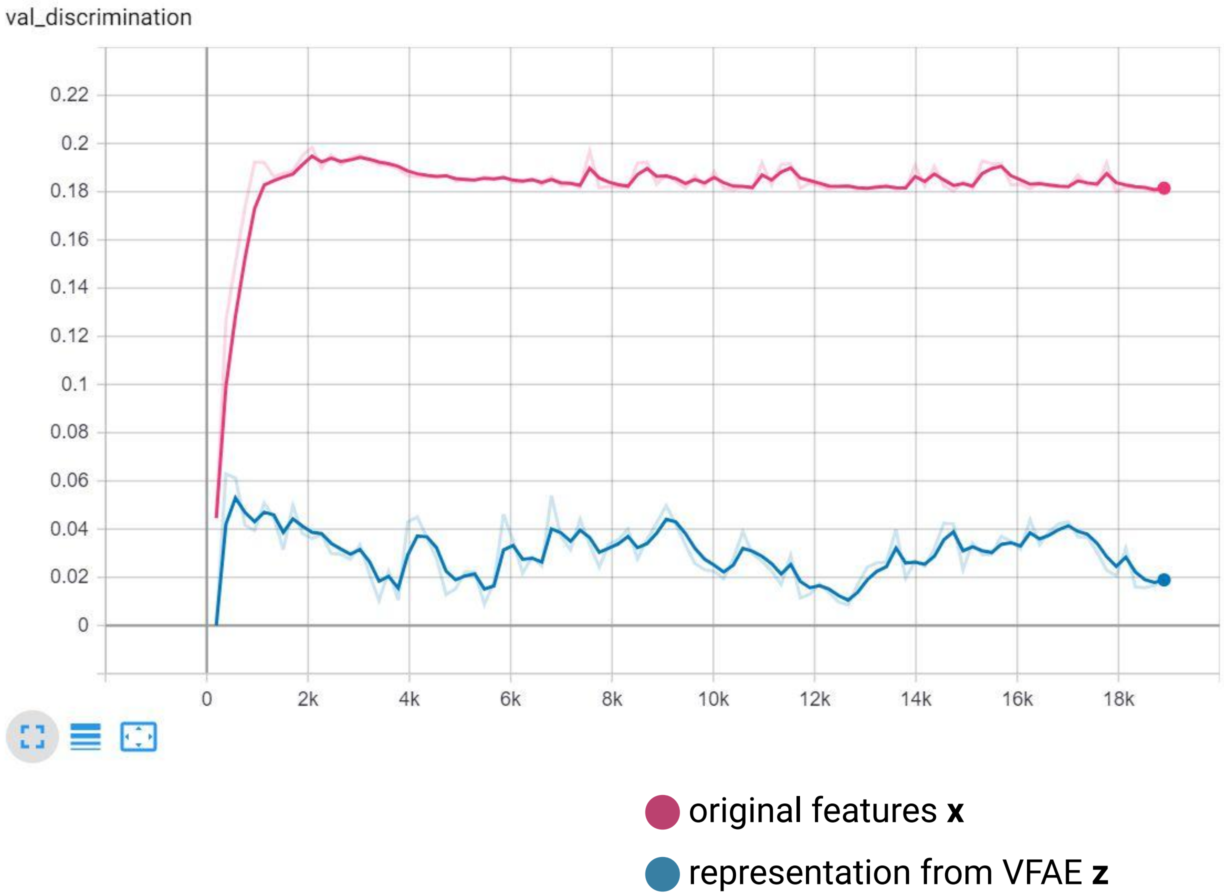
$$\text{Discrimination} = \left| \frac{\sum_{n=1}^{N} \mathbb{I}[y_n^{s=0}]}{N_{s=0}} - \frac{\sum_{n=1}^{N} \mathbb{I}[y_n^{s=1}]}{N_{s=1}} \right|$$

$$\text{Discrimination prob.} = \left| \frac{\sum_{n=1}^{N} p(y_n^{s=0})}{N_{s=0}} - \frac{\sum_{n=1}^{N} p(y_n^{s=1})}{N_{s=1}} \right|$$

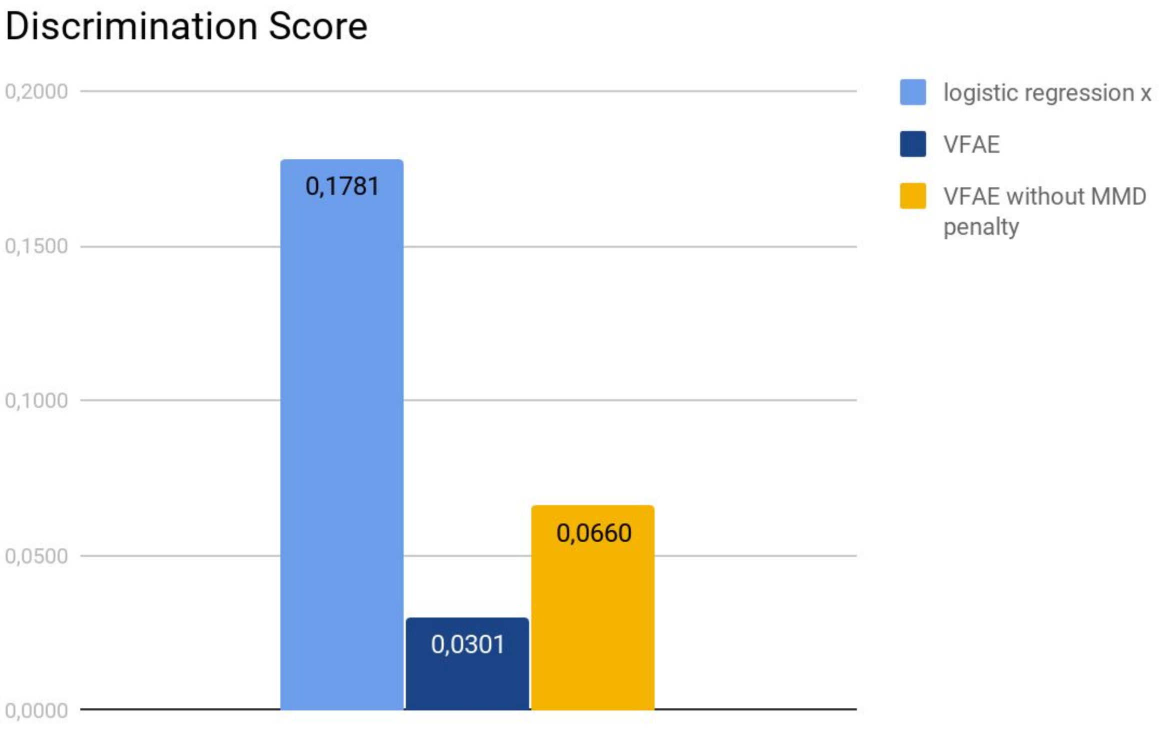
Discrimination Score

Discrimination on the validation set during training.

- discrimination on **z** stays low rather consistently **->** gender is being factored out properly
- however **z** is more noisy



Discrimination Gender



Effects of the MMD penalty

We found the use of the MMD penalty to

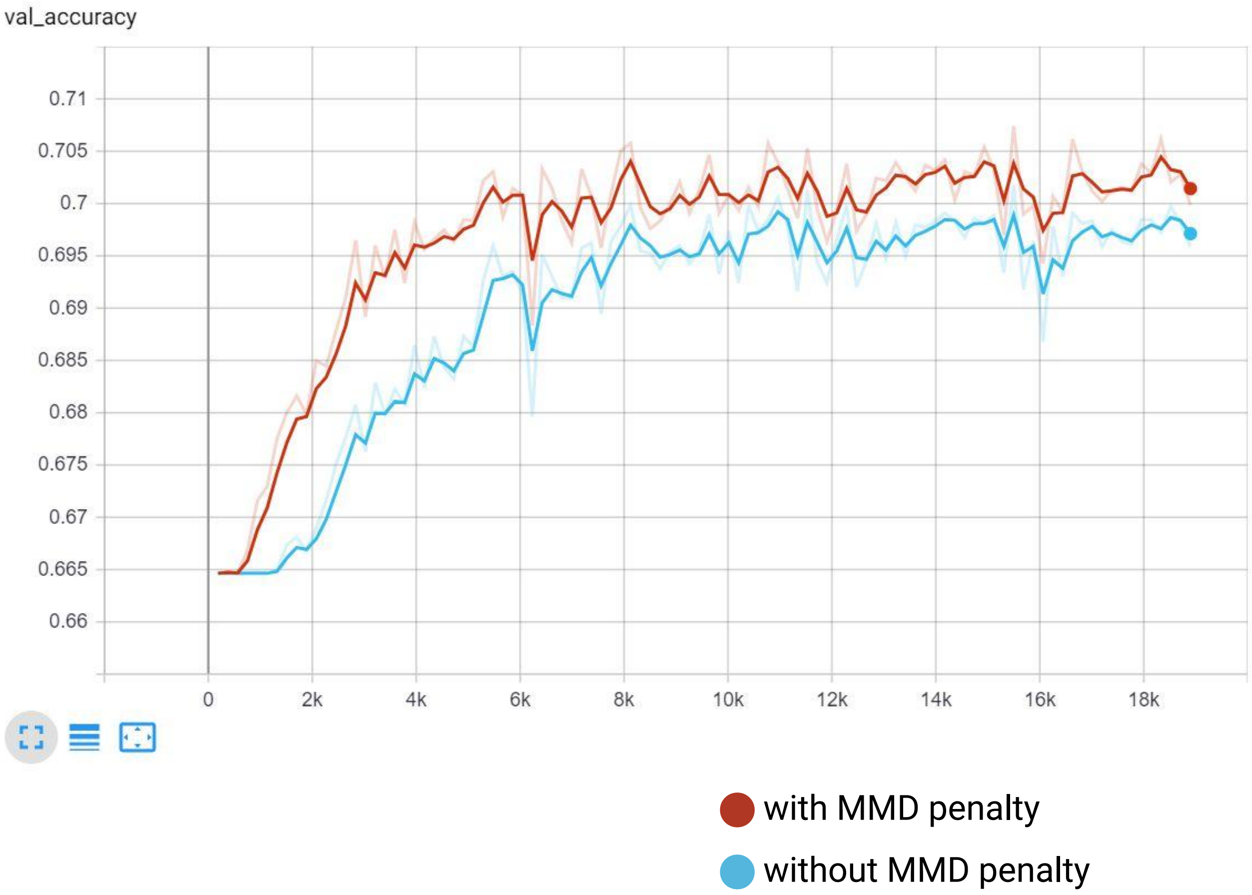
- result in higher accuracy on **s** (lower is better)
- cause high variance in discrimination during training
- make it more likely to jump into good minima

-> room for improvement

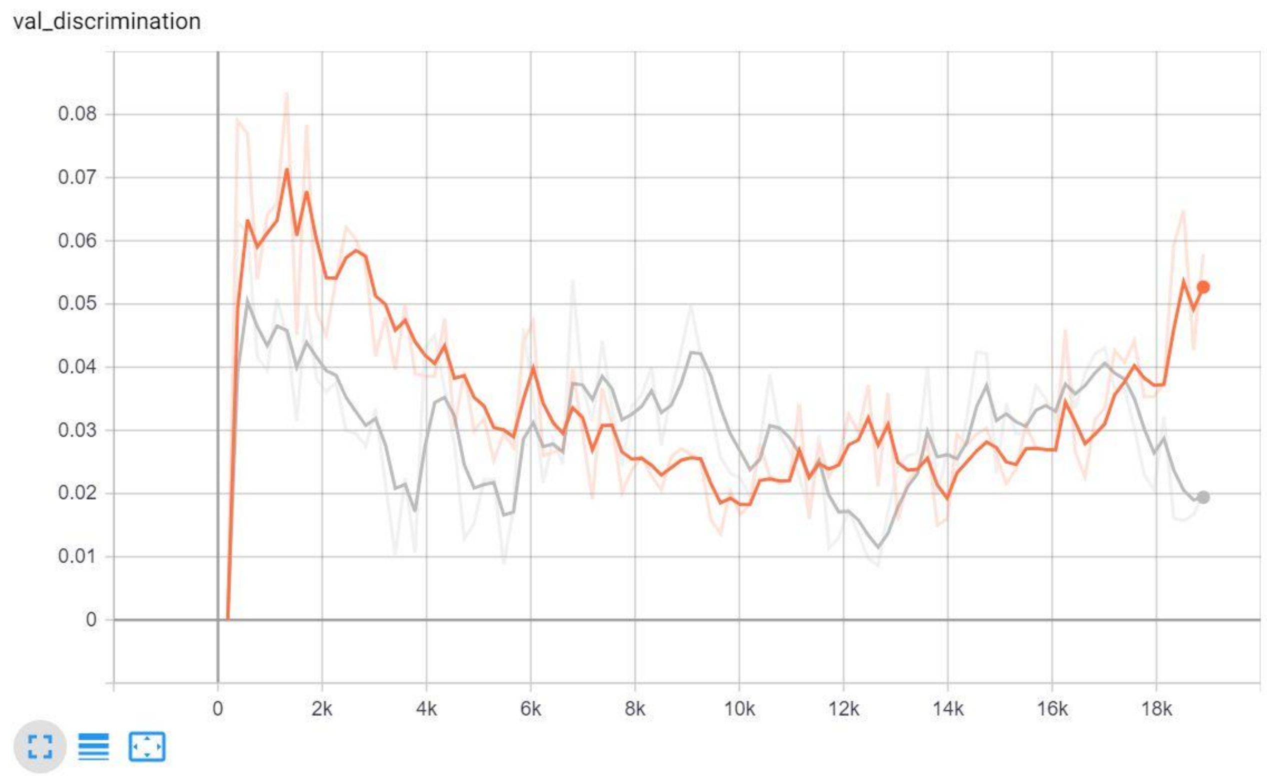
$$\ell_{\mathrm{MMD}}(\mathbf{Z}_{1_{\mathbf{s}=0}}, \mathbf{Z}_{1_{\mathbf{s}=1}}) = \| \mathbb{E}_{\tilde{p}(\mathbf{x}|\mathbf{s}=0)}[\mathbb{E}_{q(\mathbf{z}_1|\mathbf{x},\mathbf{s}=0)}[\psi(\mathbf{z}_1)]] - E_{\tilde{p}(\mathbf{x}|\mathbf{s}=1)}[\mathbb{E}_{q(\mathbf{z}_1|\mathbf{x},\mathbf{s}=1)}[\psi(\mathbf{z}_1)]]\|^2$$

Predicting **s** without MMD

- MMD penalty resulting in higher accuracy on **s**
- could be due to:
  - different scaling
  - longer training time



Discrimination with MMD more noisy



Improvement Ideas

Improvement Ideas

- actual MMD instead of the fast approximation
- replace MMD with a different penalty
- try common ideas for AutoEncoders:
  - add input noise for robustness
  - use Contractive AutoEncoder penalty

Thanks for listening!